THE SBC GAZETTE

A Terminal For "KAOS" (Kim, Aim, OSI, Sym

Bruce Land
Baltimore, MD

A "terminal" is what you use to send messages to the computer and to receive messages from it.

KIM and SYM have a terminal built in -- a hex keypad to send messages, and a 6-digit hex LED panel to receive them. The arrangement is simple, economical (in initial cost, at least), and slow. Sooner or later, one tires of using only the onboard hex pad and 6-character LED display, and yearns for an ASCII keyboard and CRT display.

Rockwell's AIM has a keyboard, 20-character display, and 20-character printer, and Ohio Scientific's Superboard has a keyboard, video board, and RS232 output; even so, owners of these other popular 6502 systems sometimes want an external video board to display longer lines.

Many articles on how to attach different combinations of keyboards and displays have been published. Let's look at some of the pros and cons of different systems, and then at the one I chose. I believe the one I chose is, for a one-board system, about the most cost-effective method of obtaining a very versatile ASCII-plug keyboard input and a memory-mapped video output to a CRT display.

Of all the ways to obtain ASCII I/O, the simplest and perhaps the cheapest is to use a parallel-connected keyboard and a video RAM display. Hal Chamberlin, in "Software Keyboard Interface with a Pittance of Hardware" (Kilobaud, January 1978), discusses how to install an unencoded keyboard as a software scanned device connected to a PIA-type parallel input port. This uses a minimum of hardware, and not much CPU time. The OSI C1P and C4P, the Apple, the PET, and others use a similar method to connect their keyboards.

Chamberlin gives complete schematics and KIM software. Software for other 6502 systems would be very similar. The hardware will work with any port and should cost less than $30.

Don Lancaster announced the first KIM pseudo "video RAM" in Kilobaud (June 1977) and in Popular Electronics (July 1977). Complete schematics were published, and some software. Kits were marketed for about $35 by PIA Electronics, Inc., 1020 W. Wilshire Blvd., Oklahoma City, OK 73116. This system relied on the CPU to run the display, and while the CPU was busy elsewhere the video was blank. For continuous display it was necessary to write software to have the CPU maintain the display and run the program at the same time. A foreground/background type of operation is needed, and this can get quite complicated.

The amount of CPU time required for the Lancaster display varies, but you can get an idea from the hex keyboard scan and display of the basic KIM. There, about 20% of the CPU time is spent on I/O software. To use the Lancaster system, decide how much delay you can tolerate in keyboard response, how long you want to display, and how often you will scan the keyboard for an entry -- five times a second, ten, or more -- and write your software accordingly.

Anything you store in a true video RAM memory location will be output as a composite video signal and displayed. The display is refreshed with TTL logic, not CPU time. A software-scanned keyboard and a video RAM are the fastest way to make an entry and get an ASCII character displayed. A video RAM is about the only practical way to do animated graphics.

M.T.U., P.O. Box 12106, Raleigh, N.C. 27605, now sells a true video RAM for approximately $300, assembled and tested. The M.T.U. board has 320 X 200-bit resolution (64,000 bits, or about 8K of RAM), which is the highest I have seen.

The big disadvantage of a video RAM driven CRT display is the lack of softwre compatibility. Almost all, maybe 95% of the software published for KIM, AIM, or SYM, is built to run with the respective ROM-based monitor program. That means you will have to rewrite the I/O of the software to run with a parallel keyboard and a video RAM. If you expect to write or adapt most of your software, then this method is very attractive; if you don't want to write a lot of special I/O programs, you should think twice before going this way.

A "6502 Video Driver Routine" software package is available for KIM from Forethought Products, 87070 Dukhobar Rd., Eugene, OR 97402, (503) 485-8575. It furnishes cursor movement, line and page functions, scrolling, etc., and should save the good programmer some time. Video RAM cards are made by several other manufacturers: Matrox

(5800 G Andover Ave., Montreal, Quebec H4T 1H4, Canada, telephone (514) 735-1182) has several models from $225-$500; The Computerist (34 Chelmsford St., Chelmsford, MA 01824 (617) 256-3649) has one for $245.

You want hard copy? A popular hard-copy output device is a teletype, known to several generations of ham radio operators as a TTY. KIM, AIM, and SYM have built-in monitor routines for TTY's and other serial devices. (I get tired of writing KIM, AIM, or SYM. We need a symbol to refer to all three systems. Try KAS. Or we could add OSI, another popular 6502 system, and call it KAOS, pronounced "Chaos.")

A used TTY sells for $500 up, and will furnish readable, dependable, noisy, all-caps, 110-baud output. A TTY may also have a paper tape reader and punch for mass storage, but don't bother with it. The KAOS cassette tape storage is quieter, more reliable, and faster. The graphics capabilities of a TTY are very limited.

Other printers are available with parallel or serial I/O, grpahics capabilities, upper and lower case, and better print quality. Of course, they usually cost more. Among them are Centronics terminals, the Texas Instruments Silent 700, Decwriters, Diablo, Qume, etc.

The great advantage of a serial terminal is that it works directly with the KAOS ROM's; no RAM is required to run it, and software purchased for any of the KAOS systems will run as a "black box" --just hook it up (which brings to mind the simplicity of this operation for a serial device: only three wires are needed. Hook up signal in, signal out, and ground, and you're ready to go.)

Hard copy output is a real plus, but there is something anomalous in paying three or four times as much for a printer as for the computer that drives it. Anomalous it may be, but a $3000 Diablo dances nicely to the tune played by a KIM that only cost $245 four years ago when it was shiny and new.

A video terminal such as the Lear-Siegler ADM-3 has all the serial advantages of a TTY, but no hard copy. Telecommunications, Alexandria, VA 22303, (703) 683-4019, sells rebuilt Datapoint video terminals for $500 up. New terminals can be found from $750 to $3000. But why buy a $750 terminal for a $180 CPU? You can buy a complete PET or OSI computer for not much more!

The answer, of course, lies in your purpose. If you're going to use the computer occasionally, for no more than a few hours a day, then limited line length and readable print quality may be all you need. On the other hand, if you're going to do extensive word processing or software development, and will be looking at the display for hours at a time, you may be willing to pay a lot more for a sharper, cleaner display, with 80-character lines.

Even if hard copy will be needed at some stage, there are advantages in being able to work with a CRT screen up to the point of the print-out. A CRT screen displays text much faster than a TTY, displays it quietly, and does not waste paper.

The video RAM is one way to get ASCII I/O; another is with a serial video system, sometimes called a "glass TTY." This is a video board with a parallel keyboard port and a serial ASCII I/O port. You add:

(1) your own monitor, modified TV, or RF modulator with an oridnary TV'
(2) power supply;
(3) parallel ASCII keyboard;
(4) and some kind of case.

Now you have a serial video terminal with capabilities similar to those of the ADM-3.

These video boards come in all stages of completeness, price, and features. You can get RS232 or 20ma I/O; 32, 64, or 80 characters per line; upper case only, or up to 128 ASCII characters; all-TTL logic (a very old design), or ROM-based CPU; slow or fast; with or without keyboard; one to three power supplies required; $150 to $500. Be careful in your choice; a high price may signify an old, expensive design.

Any of the KAOS machines will think this kind of terminal is an ordinary serial terminal, and most software will run without any modification. Such systems can be purchased from many suppliers. I know of these:

Electronic Systems, San Jose, CA 95151 (408) 448-0800 ($200 for kit; keyboard needed);

Xitex Corp., 9861 Chartwell Drive, Dallas, Texas 75243, (214) 349-2490 ($175 kit; keyboard needed; or $375 for full kit including keyboard, case, etc.);

Electrolabs, Box 6721, Stanford, CA 94305, (415) 321-5601 ($239 A & T; keyboard needed);

Mostek Corp., 1215 W. Crosby Rd., Carrollton, TX 75006 (214) 242-0444 ($195; keyboard needed;)

Synertek Systems (who also make SYM), Box 552, Santa Clara, CA 95052 (408) 988-5600 ($389 - $450 complete);

Riverside Electronics Design, 1700 Niagara St., Buffalo, N.Y. 14207 (716) 875-7070 ($225 A & T; keyboard needed; $150 complete kit including keyboard.)

Netronics R & D, Ltd., 333 Litchfield Rd., New Milford, CT 06776, (800) 243-7428 ($149.95 + $3 postage.)

After much looking and reading, and several long-distance telephone calls, I chose the "Stand Alone ASCII/Baudot Computer Terminal" by Netronics R & D, Ltd. This unit will provide 64 or 32 characters per line -- 64 for TV direct or video monitor, and 32 for use with a modulator and plain TV. The baud rate is 110 or 300 ASCII, 45.45 or 74.2 Baudot. Output is either RS232 or 20 ma current loop (TTY "similar"). All printable ASCII characters are available (upper and lower case) as well as 32 special characters (Greek letters, symbols, superscripts, and graphic characters).

Complete cursor control is provided, including absolute and relative X - Y addressing. This allows low-resolution graphics and computed relative cursor jumps. At 300 baud you cannot do animation.

The Netronics video board has an on-board +5V regulator, and draws about 450 ma. If it is used with their keyboard, you supply +8VDC (or +5VDC) at 500 ma and 6.3VAC at about 50 ma (most keyboard inverter chips require -12V DC; the Netronics circuit eliminates the need for this supply. It uses a voltage doubler to convert the 6.3VAC to -12VDC for the keyboard encoder chip and the RS232 I/O levels.)

The video board mounts underneath the keyboard and both fit into the Netronics $20 keyboard case, leaving room for the necessary transformers and capacitors. When the keyboard and the video board are assembled and housed in the case, they provide full ASCII or Baudot input with some interesting extras, and everything needed for the output display except a monitor.

The Netronics documentation is a little on the light side; nevertheless, assembling the kit should be relatively easy for anyone with kit-building experience. The copper traces and pads are very small, so a small-tip, low-wattage soldering iron is a must. Take your time, and inspect each of the more than 1000 joints for proper solder flow and absence of solder bridges. There are many plated-through jumper holes in the board, and it is easy to insert a component in the wrong hole. The component numbers are marked on the board, but the jumper holes do not have a silkscreened outline around them as Heathkit boards do. If you have any doubt about the proper placement of a component, trace the schematic and follow the foil traces. (The first-time kit builder is advised to get some expert supervision in positioning the components. It's discouraging to have to back up.)

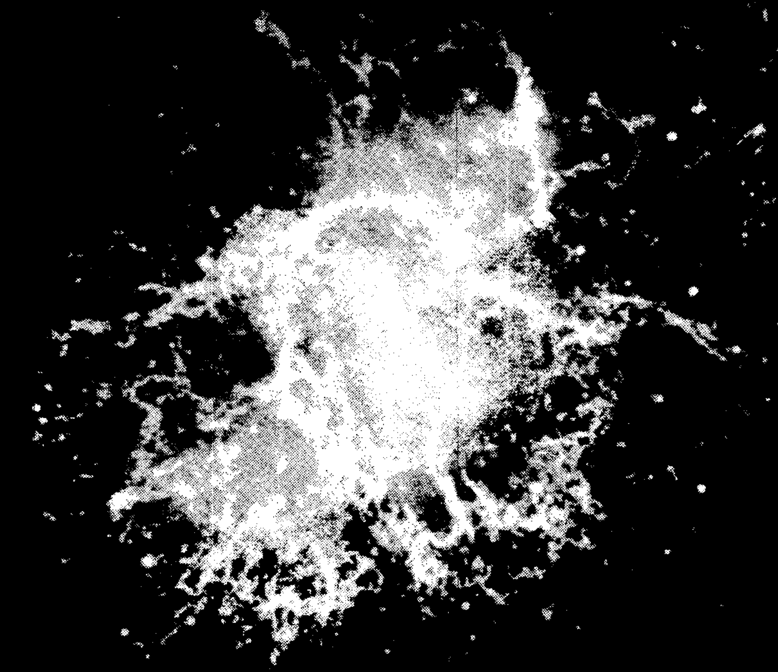
A good photograph showing correct placement of components on a completed board should be included with the documentation, but is not. The kit does not include an RS232 connector.

My group of five electrical engineers built 9 of these terminals. Five of the boards failed to work at first because of poor solder joints or misplaced jumpers. One board had a bent IC pin, and one had 3 jumpers missing. One, assembled by a good solderer with a known good board for reference, worked the first time it was hooked up.

A few modifications to the board might be considered. If you replace jumper S10 with a normally closed pushbutton switch, you can generate the BREAK command like a TTY.

Put a SPDT switch in place of J3-J4 on the keyboard, and you can switch easily between all-caps

A BRILLIANT FUTURE FOR YOUR AIM-65 WITH THE BANKER MEMORY ™



Your 36K of free address space is the AIM's most valuable and limited resource. With today's large capacity RAM boards, ROM boards, disk systems, video boards, and other expansion accessories it is easy to deplete this resource before the application requirement is satisfied. MTU has solved this problem.

THE BANKER MEMORY contains 32K of RAM, 4 PROM sockets for 2716/2732/2332, a PROM programmer, 40 bits of parallel I/O, and 4 timers from two 6522 I/O chips. Addressing is extremely flexible with the RAM independently addressable in 4K blocks, PROM's independently addressable, and I/O addressable anywhere on a 64 byte boundary (even in AIM's I/O area at AXXX by adding a single jumper to the AIM).

This may sound familiar, but read on! Unlike other AIM compatible memory boards, THE BANKER MEMORY has on-board bank-switching logic! The four 8K blocks of RAM plus the 4 PROM sockets make up 8 **resources**, each associated with a bit in an Enable Register. Through this Enable Register resources may be turned on and off under software control. When a resource is off, its address space is freed for other uses. You can even put BANKER resources at the same address and switch among them for virtually unlimited RAM and PROM expansion! You can even have multiple page zero's and stacks! Do you need 160K byte of memory? It only takes 5 of THE BANKER MEMORY boards and you end up with 5 page zeros and stacks to boot!

There's more! The BANKER MEMORY also incorporates 18 bit addressing which allows for the 256K address spaces of the future. RAM, PROM, and I/O each has its own full 18 bit address decoder which allows these resources to be in different 64K banks. This board and other MTU products, such as our 320 by 200 dot VISIBLE MEMORY and Floppy Disk Controller with 16K DMA RAM, can turn your AIM into a truly powerful 6502 computer that far surpasses the packaged systems in functional performance.

INTRODUCTORY SPECIAL K-1032-1 32K BANKER MEMORY FULLY ASSEMBLED AND TESTED $395.00 ($450.00 as of March 1, 1980) or the K-1032-2 16K RAM only with bank switching and 18 bit address bus only $295.00

Isn't it time you took a closer look at MTU — we offer you power now with an eye to the future.

WRITE OR CALL TODAY FOR OUR 48 PAGE FALL 1980 6502 CATALOG
International requests include $1.00

VISA and MASTERCARD accepted

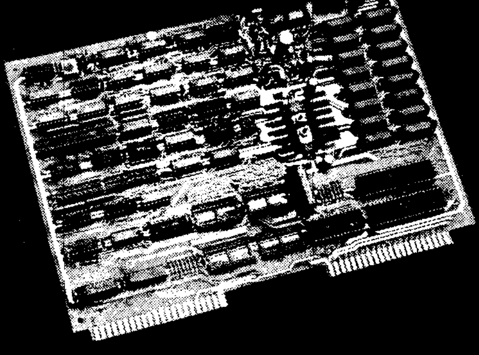
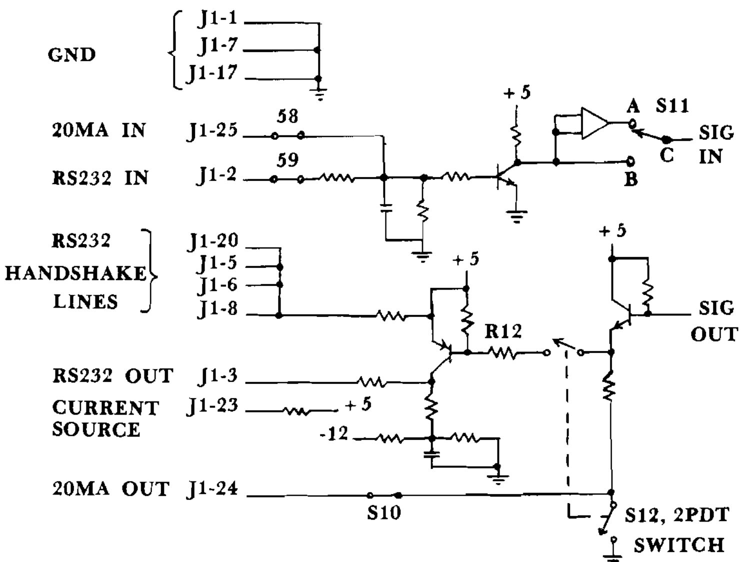


Photo credit:
SUPERNOVA CRAB NEBULA:
Palomar Observatory,
California Institute of Technology

MTU
Micro Technology Unlimited
2806 Hillsborough Street
P.O. Box 12106
Raleigh, NC 27605, U.S.A.
(919) 833-1458

with numbers (TTY mode) and typewriter mode, with both upper and lower case. When you are writing or running programs in BASIC you will probably find the all-caps mode most convenient. If you intend to do any word processing, you will find that ability to change easily to upper or lower case is very helpful.

If you think you may want to change from RS232 to 20 ma loop, install a SPDT toggle switch at S11, and a DPDT switch at S12. One pole of toggle switch S12 should be in series with R12, and the other replaces jumper S12. These switches permit you to change from one system to the other without changing 6 jumpers. (Fig. 1). For RS232, set switch S11 to position B, close S12, and use pins 2 and 3 for I/O. For 20 ma current loop, set switch S11 in position A, switch S12 open, and use pins 24 and 25 for I/O. Jumpers S8, S9, and S10 are installed as shown.



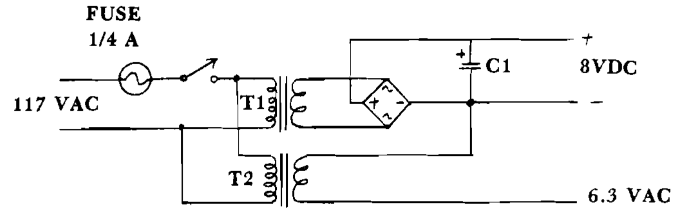
SWITCHES SHOWN IN 20 MA POSITION

I/O MODIFICATION FIGURE 1

Other lines in J1 will have to be connected to your terminal, but different computers and terminals may require different lines. KIM has a 20 ma current loop I/O, but the input to the terminal needs to be inverted. Set Switch S11 to Position B, and open Switch S12. Wire KIM A-1 to Netronics J1-1, KIM A-T to Netronics J1-24. Jumper KIM A-S (S acts as a current source) to A-U and wire A-U to Netronics J1-25. (You can use Netronics Pin J-1 jumpered to J1-25 as a current source; if you do this, then do not use KIM A-S.)

You could make a simple RS232 adapter for KIM to talk over, but that is another story. Pins J1-1-5, 6, 8, 17, and 20 are handshake lines for talking to a modem, and will not be used by KIM.

The power supply shown in Fig. 2 may not be ideal, but it works and fits inside the keyboard case.



POWER SUPPLY FIGURE 2

PARTS LIST

T1 Transformer 6.3 VAC @ 1.2 A Radio Shack #273-050 $3.49
T2 Transformer 6.3 VAC @300 ma Radio Shack #273-1384 2.49
D1 Diode bridge, 1A, 50 PIV Radio Shack #276-1161 .79
6 Capacitor, 3300 uf, 35V Radio Shack #272-1021 2.99

The Netronics kit has a few bad features. One is the lack of enough detail in documentation. The next may be only a personal idiosyncrasy, but I strongly prefer to use a complete set of IC sockets; Netronics provides sockets only for the 24- and 40-pin IC's.

I wish they had provided an RS232 chassis connector -- perhaps even as an option -- so I wouldn't have had to order one from another company.

The printed circuit board for the Netronics keyboard is a little flimsy for key pounding. If it is mounted properly it is perfectly OK, but the mounting instructions are included only with the optional case, not with the keyboard itself.

There is no line feed key; Control J yields a line feed. If your computer echoes a line feed when you send it a carrige return, you're okay; otherwise you have either a programming problem or a minor pain in the neck.

No serial video board I have seen -- Netronics included -- has high-resolution graphics like a memory-mapped video board. This could be provided with a RAM character generator, but it really isn't expected at this low price. The Netronics 20 ma current loop is not isolated like the Xitex, and so may not work well with some devices. It does not work well with all the devices I have tried, including KAOS systems.

Granted these deficiencies, why am I glad I bought the Netronics? To summarize:

Quick delivery via an 800 phone number and credit card.

Complete cursor control.

TTY mode, with upper/lower case eaisly available.

The full ASCII character set plus the Greek alphabet, other characters, and some graphic symbols.

Shift lock, control key, and escape key.

A true delete key (Some delete keys only back up the cursor; this one also erases the unwanted character.)

The board works directly with my KIM TTY monitor ROM -- no special software support.

My KIM now has a video terminal which cost less than the KIM. It is a complete, working terminal

which will talk not only with KIM but also with time-sharing systems anywhere. I consider it a very efficient and cost-effective means of obtaining ASCII input/output for any of the four KAOS systems.

Given the delay between writing and publication, by the time you read this there may be something better and/or cheaper on the market. These comments should help you to analyze the data sheets and schematics. I can testify that a careful kit-builder, in a few evenings of work, can put together a very attractive and efficient terminal at a very reasonable price. ©

SYMple Clock

A. M. MacKay
Owen Sound, Ontario

There are lots of clock programs around, but this one is a little different, and a lot more useful than most. It is written for a SYM-1 with 4K memory, but will work with a bare board. It is similar in some ways to the one in Issue 1 of Compute II, but doesn't require Basic or a CRT.

While most clock programs using the LED readouts won't do anything else while the clock is running, this one sits at the top of your memory, out of the way of most programs, and quietly ticks away while you and your SYM do other and better things. But when you want the time, either visually or for use by a program, just call it and there it is.

Since a program such as this will likely be stored on tape and loaded when required over the years, no page zero slots are used so you don't have to remember or keep track of them.

The clock runs in 24 hour format. If you prefer 12, change location 0FAB to 12. Then to start the clock, enter the hours, minutes and seconds, with fifteen or twenty seconds lead time, into locations 0FFD, 0FFE and 0FFF respectively. Then enter "GO F3F" and at the exact second hit "CR" and presto! Nothing happens! Ah, but it does. Your clock is running, quietly minding it's own business, eagerly awaiting your summons. Now, to see the time, all you do is hit "SHIFT CALC CR" or "SHIFT 0 CR" or any other "UNRECOGNIZED" command. The time will be displayed for a few seconds, then the readouts will be blanked except for a row of dots. As soon as you see the dots, you can go back to whatever you were doing with your SYM. If you want the time displayed during, and as part of, a program, just use "JSR B9 0F" and there it is.

Most importantly, though, if you want the time for controlling purposes, just call it at 0FFD, 0FFE and 0FFF with your program. It can, at the proper time, sound an alarm, turn off the lights, turn on your lights, and/or whatever makes you happy.

If you don't want clock time, but just the time since your SYM was turned on (actually since the clock started), don't enter anything in 0FFD-F. Just hit "GO F3F CR" and the clock will automatically start at 00 hours 00 minutes 00 seconds.

If your SYM is new, this is a good chance to experiment, changing things to suit your purposes. For example, try moving "DLY" from line 1090 to line 1010 (change "EB" to "D4" at location 0FEA). Your SYM now looks like a cheap digital clock. Now try changing "0A" at location 0FD2 to "1A", then move lines 1070 and 1080 to a new location between lines 1000 and 1010. Do you prefer the display this way?

The theory of operation is similar to that given for my clock article in Compute II no. 1, which required Basic and a CRT. However, the program is somewhat different because the clock in that article kept time in hex, while this one keeps time in decimal. To work in decimal with "SED" you must use ADC or SBC. "INC" just doesn't work.

Your SYM-1 is very powerful by itself, and is the basis for an extremely complex and powerful system. To get the most from it, I urge you to join the SYM-1 Users' Group, P.O. Box 315, Chico, CA 95927. And, of course, subscribe to and keep reading COMPUTE!

```
0005              .OS
0010 ;       ****************************************
0020 ;       ****************************************
0030 ;       ***                                  ***
0040 ;       ***    > > > SYMPLE   CLOCK < < <    ***
0050 ;       ***                                  ***
0060 ;       ***    BY A. M. MACKAY               ***
0070 ;       ***    CO-ORDINATOR, SURVEY DEPT.    ***
0080 ;       ***    GEORGIAN COLLEGE A.A.T.       ***
0090 ;       ***    1150 EIGHTH STREET EAST       ***
0100 ;       ***    OWEN SOUND, ONTARIO CANADA    ***
0110 ;       ***    N4K 5R4       JUNE 18, 1980   ***
0120 ;       ***                                  ***
0130 ;       ****************************************
0140 ;       ****************************************
0150 ;
0160 ;
0170              .BA $0F3F
0180 ;
0190 ;       * * * DEFINITION OF LABELS * * *
0200 ;
0210 OUTBYT       .DE $82FA
0220 SCAND        .DE $8906
```

```
                 ACCESS    .DE $8B86
                 IFR2      .DE $A405     ;FLAG FOR DISPLAY TIMER
                 DISBUF    .DE $A640
                 URCVEC    .DE $A66D
                 IRQVEC    .DE $A67E
                 CLRINT    .DE $AC04
                 TICH      .DE $AC05
                 TILL      .DE $AC06
                 ACR       .DE $AC0B
                 IFR       .DE $AC0D
                 IER       .DE $AC0E
                 ;
                 ;         * * * INITIATE TIMER * * *
                 ;
0F3F- 20 86 8B   START     JSR ACCESS    ;UNWRITE PROTECT SYS RAM
0F42- A9 71                LDA #L,CLOCK  :SET IRQ
0F44- 8D 7E A6             STA IRQVEC    ;  VECTOR
0F47- A9 0F                LDA #H,CLOCK  :     TO
0F49- 8D 7F A6             STA IRQVEC+1  :      "CLOCK"
0F4C- A9 B9                LDA #L,TIME   ;SET UNRECOGNIZED
0F4E- 8D 6D A6             STA URCVEC    ;  COMMAND VECTOR
0F51- A9 0F                LDA #H,TIME   ;     TO
0F53- 8D 6E A6             STA URCVEC+1  :      "TIME"
0F56- A9 C0                LDA #$C0      ;SET BITS 6 & 7
0F58- 8D 0B AC             STA ACR       ;  FOR FREE RUNNING MODE
0F5B- 8D 0E AC             STA IER       ;     AND T1 INTERRUPT ENABLE
0F5E- AD 0D AC             LDA IFR       ;CLEAR T1 FLAG BIT 6 BUT
0F61- 29 BF                AND #$BF      ;  DON'T DISTURB OTHER
0F63- 8D 0D AC             STA IFR       ;     IFR BITS
0F66- A9 50                LDA #$50      ;SET
0F68- 8D 06 AC             STA TILL      ;  TIMER
0F6B- A9 C3                LDA #$C3      ;     FOR 1/20 SEC AND
0F6D- 8D 05 AC             STA TICH      ;        START TIMER
0F70- 60                   RTS
                 ;
                 ;         * * * INTERRUPT SERVICE ROUTINE * * *
                 ;
0F71- 48         CLOCK     PHA  ;        ;SAVE ACCUMULATOR
0F72- F8                   SED  ;        ;TIME IS IN DECIMAL MODE
0F73- CE FC 0F             DEC COUNT     ;SEE IF 1 SEC HAS PASSED
0F76- D0 3B                BNE EXIT      ;IF NO, EXIT
0F78- A9 14                LDA #20       ;IF YES,
0F7A- 8D FC 0F             STA COUNT     ;  RESTORE COUNT
0F7D- 18                   CLC  ;        ;     AND
0F7E- A9 01                LDA #01       ;        ADD 1
0F80- 6D FF 0F             ADC SECS      ;           TO
0F83- 8D FF 0F             STA SECS      ;              SECS
0F86- C9 60                CMP #$60      ;SEE IF 60 SECS HAS PASSED
0F88- D0 29                BNE EXIT      ;IF NO, EXIT
0F8A- A9 00                LDA #00       ;IF YES, RESET
0F8C- 8D FF 0F             STA SECS      ;  SECS TO ZERO
0F8F- 18                   CLC  ;        ;     AND
0F90- A9 01                LDA #01       ;        ADD
0F92- 6D FE 0F             ADC MINS      ;           ONE TO
0F95- 8D FE 0F             STA MINS      ;              MINS
0F98- C9 60                CMP #$60      ;SEE IF 60 MINS HAS PASSED
0F9A- D0 17                BNE EXIT      ;IF NO, EXIT
0F9C- A9 00                LDA #00       ;IF YES, RESET
0F9E- 8D FE 0F             STA MINS      ;  MINS TO ZERO
0FA1- 18                   CLC  ;        ;     AND
```

```
0FA2- A9 01      0830              LDA #01       ;      ADD
0FA4- 6D FD 0F   0840              ADC HOUR      ;         ONE TO
0FA7- 8D FD 0F   0850              STA HOUR      ;            HOUR
0FAA- C9 24      0860              CMP #$24      ;SEE IF 24 HOURS HAS PASSED
0FAC- D0 05      0870              BNE EXIT      ;IF NO, EXIT
0FAE- A9 00      0880              LDA #00       ;IF YES, RESET
0FB0- 8D FD 0F   0890              STA HOUR      ;  HOUR TO ZERO
0FB3- AD 04 AC   0900 EXIT         LDA CLRINT    ;ENABLE TIMER INTERRUPT
0FB6- D8         0910              CLD  ;        ;BACK TO HEX
0FB7- 68         0920              PLA  ;        ;RESTORE ACCUMULATOR
0FB8- 40         0930              RTI
                 0940 ;
                 0950 ;          * * * DISPLAY ROUTINE * * *
                 0960 ;
0FB9- 20 86 8B   0970 TIME         JSR ACCESS    ;UNWRITE PROTECT SYS RAM
0FBC- 48         0980              PHA  ;        ;SAVE ACCUMULATOR
0FBD- 8A         0990              TXA  ;        ;  AND
0FBE- 48         1000              PHA  ;        ;     X-REGISTER
0FBF- AD FD 0F   1010              LDA HOUR      ;PUT
0FC2- 20 FA 82   1020              JSR OUTBYT
0FC5- AD FE 0F   1030              LDA MINS      ;  TIME ON
0FC8- 20 FA 82   1040              JSR OUTBYT
0FCB- AD FF 0F   1050              LDA SECS
0FCE- 20 FA 82   1060              JSR OUTBYT    ;     DISPLAY
0FD1- A9 0A      1070              LDA #$0A      ;SET NUMBER OF
0FD3- 8D FB 0F   1080              STA CNT1      ;  TIMEOUTS FOR DISPLAY
0FD6- A9 FF      1090 DLY          LDA #$FF      ;SET LENGTH OF
0FD8- 8D 1F A4   1100              STA $A41F     ;  TIMEOUT
0FDB- 20 06 89   1110 DISPL        JSR SCAND     ;LIGHT LEDS
0FDE- AD FA 0F   1120              LDA MASK      ;CHECK TIMER
0FE1- 2C 05 A4   1130              BIT IFR2      ;  IRQ
0FE4- 10 F5      1140              BPL DISPL     ;IF NO IRQ REPEAT
0FE6- CE FB 0F   1150              DEC CNT1      ;ELSE START AGAIN
0FE9- 10 EB      1160              BPL DLY       ;FINISHED?
0FEB- A2 05      1170              LDX #$05      ;CLEAR
0FED- AD FA 0F   1180 CLR          LDA MASK
0FF0- 9D 40 A6   1190              STA DISBUF,X  :  DISPLAY
0FF3- CA         1200              DEX
0FF4- 10 F7      1210              BPL CLR
0FF6- 68         1220              PLA  ;        ;RESTORE
0FF7- AA         1230              TAX  ;        ;  X-REGISTER AND
0FF8- 68         1240              PLA  ;        ;     ACCUMULATOR
0FF9- 60         1250              RTS
                 1260 ;
                 1270 ;          * * * STORAGE DEFINITIONS * * *
                 1280 ;
0FFA- 80         1290 MASK         .BY %10000000          ;BIT 7 ONLY
0FFB-            1300 CNT1         .DS 1         ;PROVIDE SPACE FOR CNT1
0FFC- 14         1310 COUNT        .BY 20        ;SET COUNT TO 20
0FFD- 00         1320 HOUR         .BY 00        ;START TIME AT 00 HOURS
0FFE- 00         1330 MINS         .BY 00        ;              00 MINUTES
0FFF- 00         1340 SECS         .BY 00        ;              00 SECONDS
                 1350              .EN
LABEL FILE:  [ / = EXTERNAL ]

/OUTBYT=82FA     /SCAND=8906     /ACCESS=8B86
/IFR2=A405       /DISBUF=A640    /URCVEC=A66D
/IRQVEC=A67E     /CLRINT=AC04    /TICH=AC05
/TILL=AC06       /ACR=AC0B       /IFR=AC0D
/IER=AC0E        START=0F3F      CLOCK=0F71
EXIT=0FB3        TIME=0FB9       DLY=0FD6
DISPL=0FDB       CLR=0FED        MASK=0FFA
CNT1=0FFB        COUNT=0FFC      HOUR=0FFD
MINS=0FFE        SECS=0FFF
//0000,1000,1000
>
```

Expanding KIM-Style 6502 Single Board Computers

Hal Chamberlin

Editor's Note: Hal ended his first installment with this . . .

"The real question at this point then is: How many expansion boards can the unbuffered microprocessor bus drive before becoming overloaded? The 6502 microprocessor is rated to drive slightly more than 1 standard TTL load (equivalent to five low power shottky loads) on its address and data busses while most of the RAM's and ROM's tied to the data bus can drive two standard TTL loads. The 6520, 6522, and 6530 I/O chips have the same drive capability as the microprocessor. Thus in general the answer is at least four boards provided that the expansion boards themselves buffer the bus such that only one low power shottky load (.36MA in the zero state) is presented to the bus by the board. Many boards on the market and particularly those designed for an unbuffered bus do this. Actually, any well designed board would be expected to buffer the bus in order to provide clean signals for the remainder of the board logic. The reason that only four boards can be driven instead of five is that some of the address lines are loaded by a low power Shottky decoder IC on the computer board itself.

Part 2 of 3
The Great Experiment

Of course loading the microprocessor with a full five loads puts the system right at the limit of rated drive current. One of the problems with testing digital circuitry is that there is no obvious indication of marginal operation that may later develop into a full fledged failure as components age. In order to determine the actual drive limit, the author took a fully stuffed AIM-65 (4K on-board RAM, assembler ROM and BASIC ROM's) and started adding Micro Technology K-1016 16K memory boards, the idea being to add boards until failure due to bus overload occurred. These boards use low power Shottky buffers onboard so each one would be expected to add a .36MA load to the bus.

Since the AIM's 40K of free addresses would only accomodate two of these boards, the most significant address bit was cut away from the bus at each socket position and instead connected to parallel output bits on the AIM's application connector. The boards were then jumpered to respond to addresses between 2000 and 5FFFF (hex). By programming only one output bit to be low at a time, a rudimentary bank switching setup was implemented. When the system was reset, all output bits automatically go high thus disabling all of the boards and preventing interference with the AIM monitor (since A15 was ignored, an enabled board would also respond to A000-DFFFF). A proper bank switch setup would have required a two-input OR gate (negative AND) to be tied to each of the A15 pins. In any case, it was adequate to run a memory test program.

The first trial was to install 4 of the 16K boards which worked fine as expected. Next, another card file was placed below the first and jumper wires added between the two motherboards. This gave a total of 9 bus slots which were filled with 16K memory boards. Again the memory test program (which wrote all 144K of memory with random data before reading any of it back) indicated no problem and the AIM monitor and BASIC continued to work flawlessly. A check with an oscilloscope revealed minimal signal degradation.

Finally, a third card file was added and bus jumpers installed to give a total of 14 slots. Three additional 16K memory boards were scrounged (I had no idea that more than 9 or 10 boards could be driven) to give a total of 192K of RAM. Again there were no obvious problems and the bus was being loaded to three times rated capacity! Figure 3 shows what the stack of card files looked like which is obviously impractical unless one cuts a hole in the tabletop to let the two extra card files hang below (I simply sat on a drafting stool to use the system). The rear view in figure 4 shows the interconnected motherboards and individual Board Enables from the application connector. Note the gridwork of copper braid between motherboards which makes the groundplane essentially continuous between the motherboards.

Photographs of the address and data bus signals were taken while running the memory test program and are shown in figure 5. About the only visible loading effect on the address bus is a long tail on the zero-to-one transition during phase 1 of the clock. The data bus appears to be even cleaner with just a shade over 100NS required for the data to stabilize after the leading edge of phase 2. The microprocessor was driving the data bus for the data bus for this photo (scope synced to read/write line on the bus). The zero logic levels, which one would think show the effect of gross overloading most, were still in the 0.3 volt range although the one levels were down to only 3 volts from a normal lightly loaded value of nearly 4 volts. Note the almost complete absence of noise. These "overloaded" signals actually look far better than most S-100 bus signals!

While these results are encouraging and certainly show that a four board load does not bring a system to the brink of failure, it does not mean that loading rules can be disregarded altogether. Some AIM's, as well as SYM's and KIM's, can be expected to have a weak component on-board that may not be able to drive a 12 board load adequately for reliable operation. Thus the "official" recommendation is to stick with the spec book and limit unbuffered systems to four boards. However, individual hobbyists should be able to go one or two boards over the limit with little probability of problems. Actually, addressing limitations are more likely to limit system size than bus drive capability with today's dense boards.

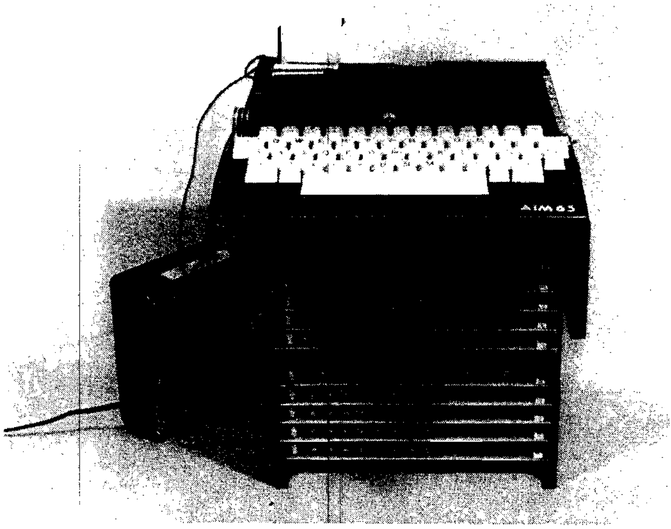


FIG.3.FRONT VIEW OF 192K RAM TEST SYSTEM

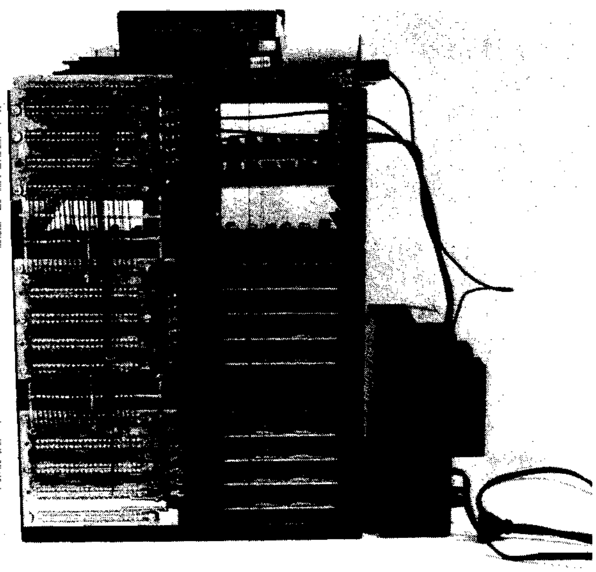
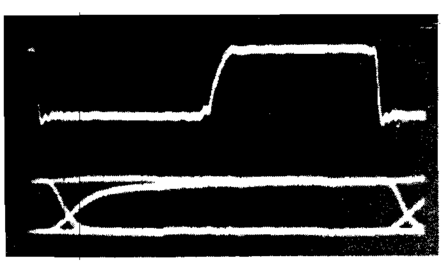
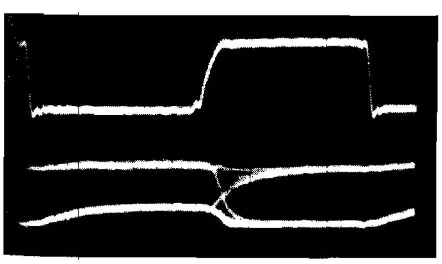


FIG. 4. REAR VIEW OF 192K TEST SYSTEM SHOWING MOTHERBOARDS WIRED TOGETHER



A. ADDRESS BUS



B. DATA BUS

FIG. 5. BUS SIGNAL WAVEFORMS IN 192K TEST SYSTEM. TOP WAVEFORM IN EACH PHOTO IS PHASE 2 CLOCK.

©

Load And Save KIM BASIC Programs On Your SYM

George Wells
LaVerne, CA

The SYM and KIM microcomputers are close cousins. Their hardware and tape interface compatibilities are well known. Not so well known is the fact that the BASIC interpreters on the two systems use the same tokens for their reserved keywords which makes transfer of BASIC programs between the SYM and KIM almost trivial.

Mike Hanna, a friend of mine who has had a KIM with BASIC for much longer than I have had my SYM with BASIC, has offered to share his library of BASIC programs with me. We had considered implementing a telephone/modem interface to accomplish this transfer but after comparing the disassembly listings of the two interpreters we decided a tape transfer would be easier. The scheme we finally settled on allows the SYM to create and read tapes in the original low speed KIM format since the SYM does not support any of the faster versions. Going from the SYM to the KIM is particularly simple; going the other way requires a short BASIC program (see listing).

SYM To KIM Transfer

STEP 1: Load the BASIC program to be transferred into the SYM.

STEP 2: Exit BASIC and return to the Monitor (by way of reset, for example).

STEP 3: Determine the end of the BASIC program by examining the two-byte pointer stored at $7D/$7E by entering .V 7D-7E. The SYM will respond with:

007D uv wx,yz

where wxuv is the end of the program (qrst) plus one. The monitor will calculate qrst for you if you can't do it in your head by entering .C wxuv-1.

STEP 4: Save the program on tape in KIM format by entering:

.S1 1,201-qrst

where qrst is the value from STEP 3.

STEP 5: Load the program into KIM BASIC in the normal manner.

KIM To SYM Transfer

In order to load KIM formatted BASIC programs into your SYM you will need to have a copy of the KIM BASIC PROGRAM LOADER listed with this article. Save this program on tape (in high speed format, of course) so that you will have it whenever you need it. NOTE: This program will not work with Monitor Version 1.0 which has an error in the KIM Load routine.

```
LIST: REM    KIM BASIC PROGRAM LOADER

 100 A=USR(&"8B86",0)
 110 A=42572: POKE A,1: POKE A+1,2: POKE A+2,255
 120 FOR I=0 TO 29
 130 POKE 300+I, PEEK(35960+I)
 140 NEXT I
 150 POKE 330,96
 160 PRINT "AFTER 'LOADED' MESSAGE, ENTER:"
 170 PRINT "POKE 125,PEEK(254): POKE 126,PEEK(255): CLEAR"
 180 PRINT USR(300,&"C6C5",&"8CAC",0)
OK
```

STEP 1: On the KIM, save the program to be transferred in the normal manner; but make sure it is saved at the original tape low speed.

STEP 2: Initialize BASIC on your SYM and LOAD and RUN the KIM BASIC PROGRAM LOADER.

STEP 3: Play the tape with the KIM program in your recorder. If you have implemented a second cassette control for your read-only recorder you will have to over-ride it since this program will only activate the original cassette control.

STEP 4: After the LOADED message, enter the command printed by the program and then SAVE a copy of the KIM program in high speed format. In case you get a BAD LOAD message, start over again at STEP 2.

SYM/KIM BASIC Incompatibilities

The obvious hardware related incompatibilities due to different address availability in the two systems require careful use of the PEEK, POKE and USR commands. Of course, different terminals may also have special requirements for cursor controls or graphics capabilities. Not so obvious are the following additional potential problem areas.

GO: SYM treats GO as a reserved word so don't enter GOTO as two words. Also make sure that GO does not appear in any variable names such as DRAGON.

GET: SYM does not implement this function but it does reserve the same token as KIM. (See MICRO 24:15 if you want to implement GET on your SYM.)

USR: The multiple parameter versions of USR will not work on the KIM. The single parameter version will require a different set of POKE commands prior to the USR but otherwise it works the same in both systems.

& "ABCD": KIM does not support hexidecimal notation.

Theory Of Operation

The key to the success of this scheme is in the fact that the Microsoft BASIC interpreters automatically recalculate the internal line pointers whenever a BASIC program is loaded. Theoretically, this means that a program that was originally saved at one address could be loaded into a different address if the tape load routine is given the ID value of $FF and the address where the program is to be loaded. The problem is that in some versions of KIM BASIC the ID value of $FF is used to save programs which means that since the IDs match when loading, the programs get loaded into their original address instead of the new specified address. There are two ways to fix this problem. First, if you have access to the KIM you can change location $2744 from $FF to $01 before the program is saved. This is part of the sequence LDA *$FF, STA $17F9.

The second method is the one the KIM BASIC PROGRAM LOADER uses which will work with any ID. It requires making a copy of the beginning of the SYM Monitor Load routine on page one of the SYM up to the point where the ID test is made. The FOR/NEXT loop in the LOADER program copies the code between address $8C78 and $8C95 and then an RTS instruction is attached to the end ($60 = decimal 96).

The jump to continue into the Monitor Load routine is performed by an interesting technique which Hans W. Gschwind of West Germany wrote about in SYM-PHYSICS 4-20. It involves using the multi-parameter version of the USR function by pushing two return addresses on the stack so that when the first subroutine finishes it returns to the address equal to the third parameter of the USR command plus one which is the continuation point in the Monitor Load routine. The next RTS instruction encountered returns to the address of the second parameter plus one which is the normal return point for BASIC high-speed tape loads.

With this background in mind it is possible to understand the following line by line explanation of the KIM BASIC PROGRAM LOADER.

LINE 100: Calls the Monitor ACCESS routine to allow passing of tape parameters to System Ram.

LINE 110: Passes tape start address of $201 and ID of $FF to tape parameters.

LINES 120 to 140: Copies first part of Monitor Tape Load routine to page one.

LINE 150: Ends page one copy with an RTS.

LINES 160 and 170: Prints message to be entered after a good load. The command must be entered manually since the KIM BASIC program will overwrite the LOADER program.

LINE 180: Jumps to address 300 (first parameter) with Y index register equal to zero (fourth parameter) indicating KIM tape format. The RTS at address 330 jumps to address $8CAD (third parameter plus one). The RTS at the end of the Monitor Tape Load routine jumps to address $C6C6 (second parameter plus one) in the BASIC interpreter which modifies the line pointers to fit the new location in the SYM.

Conclusion

Hopefully this scheme can be used to advantage by anyone having access to both a SYM and a KIM. If you find that it just doesn't work for you, try a different tape recorder. Mike and I spent many frustrating days trying to get the SYM to KIM transfer to work and it wasn't until I used a different recorder with my SYM before we finally did have success! Now we are able to transfer our BASIC programs with ease. ©